Social Model Shaping for Solving Generic DEC-POMDPs

Pradeep Varakantham
School of Information Systems
Singapore Management University
Singapore
pradeepv@smu.edu.sg

***Abstract*—Decentralized Partially Observable Markov Decision Problem, DEC-POMDP is a popular model to represent multi-agent decision making under uncertainty. However, the significant computational complexity involved in solving DEC-POMDPs has limited their application. Recently, social model shaping (TREMOR and D-TREMOR algorithms) was introduced as an alternative to solve a sub-class of DEC-POMDPs. While scalability has been improved to even solve hundred agent problems, social model shaping has been restricted to solving a sub-class of DEC-POMDPs called Distributed POMDPs with Coordination Locales (DPCL). To that end, we make two significant contributions: (a) Firstly, we enhance the model shaping approach to solve general DEC-POMDPs where there is no restriction on the agent dependencies; and (b) Secondly, we provide theoretical justification for the model shaping heuristics. The key intuition is that not all interactions between agents occur at every time step. In addition to solving 100 agent problems in weakly coupled domains (due to extension from TREMOR and D-TREMOR), we are able to show that social model shaping achieves comparable performance to leading DEC-POMDP solvers (such as IMBDP, MBDP-OC, PBIP-IPG etc.) on tightly coupled benchmark problems.**

I. Introduction

Decentralized Partially Observable Markov Decision Problems (DEC-POMDPs) have been used to represent decision problems in domains such as disaster rescue, phenomenon observing sensor webs and autonomous rover exploration [10], [12], [14]. Unfortunately, computing optimal solutions for DEC-POMDPs is NEXP-Complete [3]. Existing research has focused on two methods to address this complexity: (a) Computing approximate solutions [11], [1], [5] for general DEC-POMDPs. Leading algorithms in this category have typically been able to solve problems with two agents. (b) Optimally or approximately solving useful sub-classes of DEC-POMDPs, such as transition-independent DEC-MDPs [2], Network Distributed POMDPs, ND-POMDPs [10], Transition Decoupled POMDPs (TD-POMDPs) [14] and Distributed POMDPs with Coordination Locales (DPCLs) [12]. While, some of these approaches scale to problems with tens of agents, they cannot solve general DEC-POMDPs.

We introduce GenTREMOR, a model shaping mechanism that (a) is applicable to general DEC-POMDPs with tight coordination between agents; and (b) has significant scalability (similar to TREMOR and D-TREMOR) due to its ability to exploit structure in sparse coordination problems. Our approach builds over TREMOR (Team's REshaping of MOdels for Rapid execution) and D-TREMOR (Distributed TREMOR), algorithms that were developed to solve large DPCLs. DPCL is a sub-class of DEC-POMDP, where observation independence and sparse coordination are assumed. GenTREMOR is based on the core intuition that even when there are many possible locales of coordination between agents, given specific policies of agents there are only few active coordination locales (CLs). The following enhancements in GenTREMOR make it suitable for solving tightly coupled DEC-POMDP problems : (a) Shaping heuristics for handling CL dependencies and observation dependence; and (b) Computation of probability of occurrence of CLs;

While the scalability of model shaping has been illustrated in Varakantham *et al.* [12] and more significantly in Prasanna *et al.* [13], there has been no justification for the shaping heuristics. In this paper, we take a step towards addressing this issue by providing a theoretical justification for the model shaping heuristics. Finally, we experimented on a set of standard DEC-POMDP benchmark problems where agents are tightly coupled. We were able to show that not only was GenTREMOR able to outperform leading approaches for solving generic DEC-POMDPs on some of the problems, it was able to achieve comparable performance on all the problems. These results validate our claim that social model shaping performs well both in tightly coordinated and sparsely coordinated problems.

II. Background

We briefly describe the DEC-POMDP and DPCL models along with the TREMOR and D-TREMOR algorithms for solving DPCLs in this section.

A. DEC-POMDP and DPCL

Decentralized POMDP is represented using the tuple of $\langle \mathbb{S}, \mathbb{A}, \mathbb{P}, \mathbb{R}, \Omega, \mathbb{O} \rangle$. $\mathbb{S}, \mathbb{A}, \Omega$ are the joint states, actions and observations over all the agents and $\mathbb{P}, \mathbb{R}, \mathbb{O}$ are the joint transition, reward and observation functions respectively. The joint functions represent the coordination that exists between agents. If there is at least one state, $s \in \mathbb{S}$ where state transition depends on the individual states and actions of the agents involved, then there is transition dependence between the agents. Similarly, we define the reward and observation dependence. Given this tuple, the goal is to maximize the overall expected reward over all agents.

Distributed POMDPs with Coordination Locales, DPCL model was introduced by Varakantham *et al.* [12] to represent problems where there are only a few scenarios in which coordination is required between agents. A coordination scenario or locale is defined using the tuple $\langle e, (s_i)_1^M, (a_i)_1^M \rangle$. It represents the decision epoch, e at which a sub-set ($M \leq N$) of agents take actions a_i in states s_i. DPCL is similar to the DEC-POMDP model, except with respect to the following aspects:

(a) It assumes a state space consisting of global states and local states, with global states representing the status of tasks, i.e. $\mathbb{S} := S_g \times S_1 \times \ldots \times S_N$ where S_n is a set of local states of agent n for $1 \leq n \leq N$ and S_g is a set of task states that keep track of the execution of tasks. However, in DEC-POMDPs there is no restriction on what a state can represent.

(b) The interactions between agents are limited to Same Time Coordination Locales (STCLs) and Future Time Coordinational Locales (FTCLs) with respect to transition and reward function. STCLs represent situations where the effect of simultaneous execution of actions by a subset of agents cannot be described by the local transition ($\mathcal{P}_n$) and reward ($\mathcal{R}_n$) functions of these agents. FTCLs represent situations where actions of one agent affects other agents at a future decision epoch. More specifically, such interactions are caused due to changes in global state by an agent.

Algorithm 1 SolveDPCL()

```
1:  $\pi^* \leftarrow$ SolveIndividualPOMDPs($\{\mathcal{M}_i\}_{i \leq N}$)
2:  $\pi \leftarrow \phi$
3:  $iter \leftarrow 0$
4:  while $\pi \neq \pi^* \&\& iter < MAX\_ITERATIONS$ do
5:      $ActiveCLs \leftarrow$ GetActiveCLs($\{\mathcal{M}_i\}_{i \leq N}, AllCLs$)
6:      for all $cl \in ActiveCLs$ do
7:          $\{val_i\}_{i \in cl.n} \leftarrow$ EvalCL($cl$)
8:          $\{\mathcal{M}_i\} \leftarrow$ ShapeModels($cl, \langle \{val_i\}, \{\mathcal{M}_i\} \rangle_{i \in cl.n}$)
9:      $\pi^* \leftarrow \pi$
10:     $\pi \leftarrow$ SolveIndividualPOMDPs($\{\mathcal{M}_i\}_{i \leq N}$)
11:     $iter \leftarrow iter + 1$
```

B. TREMOR and D-TREMOR

The goal in TREMOR [12] is to find an optimal task allocation and a policy for each of the agents to accomplish their tasks. TREMOR performs an approximate branch and bound search over the set of all task allocations by using MDP based heuristics. The actual value of a specific task allocation is computed by solving the Distributed POMDPs with Coordination Locale (DPCL) model for that allocation(Algorithm 1). Algorithm 1 provides the pseudo code for solving the DPCL model for a specific allocation. There are three crucial steps performed at each iteration (a) Compute active interactions or CLs given current policies; (b) Shape models to account for the interactions; and (c) Solve updated models to obtain new policies.

We first describe the procedure employed for computing the occurrence probability of a CL for agent i. For a $cl = \langle (e, (s_i)_1^n, (a_i)_1^n \rangle$, the probability that it would be active when executing policy π is defined as

$$Pr_{cl}^t(b) = \prod_i Pr_{i,cl}^t(b_i), \text{ where}$$

$$Pr_{i,cl}^t(b_i) = \begin{cases} b_i(s_i)\pi_i(b_i, a) & t = e\,, \\ \sum_{a \in A_i} \pi_i(b_i, a) \sum_{\omega \in \Omega_i} P_i^a(\omega|b) \cdot Pr_{i,cl}^{(t+1)}(G_i^{a,\omega}(b_i)) & t \leq e \end{cases},$$

$$G^{a,\omega}(b)(s') = \frac{1}{P^a(\omega|b)} \mathcal{O}_i(s', a, \omega) \sum_{s \in S} \mathcal{P}_i(s, a, s') b(s),$$

$$P^a(\omega|b) = \sum_{s' \in S} [\mathcal{O}_i(s', a, \omega) \sum_{s \in S} \mathcal{P}_i(s, a, s') b(s)]$$

In the above expressions, $\mathcal{P}_i$ and $\mathcal{O}_i$ refer to the individual agent models (i.e. when other agents are not present in the environment. $\pi_i(b_i, a)$ refers to the probability of executing action a in belief b_i when using policy π_i. Intuitively, these expressions are obtained by traversing through the policy tree of an agent (starting from belief b_0) and computing the probability of being in a state and executing a certain action at a decision epoch. As for the second step of shaping models to account for interactions, we will explain the details of shaping in TREMOR/D-TREMOR in the next section. Finally, solving of updated models can be performed using any of the existing single agent POMDP solvers.

By starting from individual POMDPs and incrementally modifying the model to accommodate most likely interactions, TREMOR was able to scale to problems that were not feasible with earlier approaches for solving DEC-POMDPs. D-TREMOR [13] improved significantly upon TREMOR by firstly proposing a distributed mechanism to exploit the structure and proposing approximations in identifying and evaluating CLs to improve efficiency. This enhanced social model shaping algorithm was shown to scale to problems with hundred agents while providing high quality solutions.

Figure II-B provides examples of the illustrative rescue problems employed to test D-TREMOR algorithm [13]. The goal for the rescue robots is to rescue victims (in red) while avoiding collisions with other robots in narrow corridors. The goal for the cleaner robots is to clean debris (shown as rocks) if they are obstructing rescue robots from reaching their victims. Rescue robots and cleaner robots have to accomplish their goals in the presence of transition uncertainty (uncertainty in movements) and observational uncertainty (unable to know exactly if there has been movement). Prasanna *et al.* [13] show that D-TREMOR scales

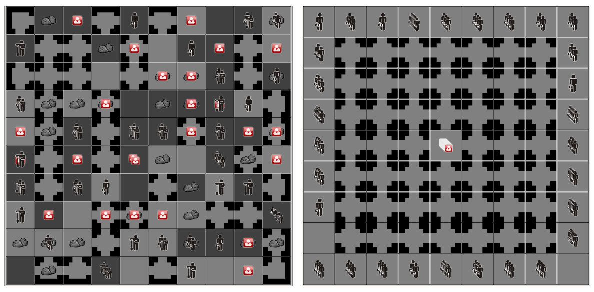

Figure 1. Maps employed to evaluate the performance of D-TREMOR

to problems with upto 100 agents in such DPCL problems (as shown in Figure II-B). Not only, was the scalability improved many folds, the solution quality was better than benchmark solution methods.

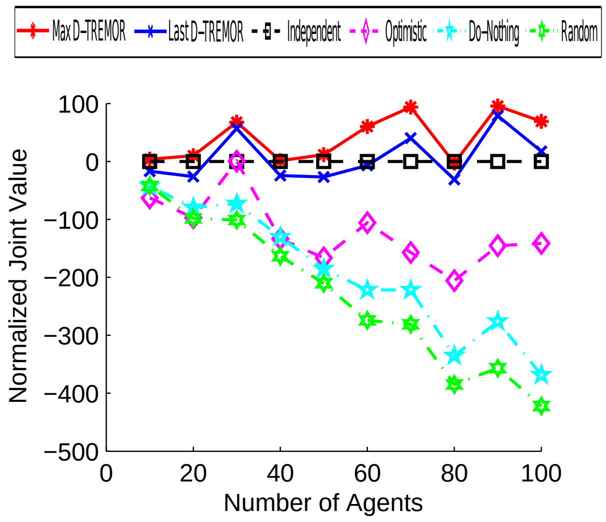

Figure 2. Scalability and solution quality results for D-TREMOR on Rescue Problem

TREMOR and D-TREMOR, however are applicable in domains where: (a) there is an explicit definition of roles or tasks to be accomplished by agents; and (b) there is observation independence, i.e. observations received by an agent are not dependent on actions taken by other agents. To address these deficiencies, we first propose a coordination locale based representation of a DEC-POMDP.

III. CL Representation of DEC-POMDP

We provide a coordination locale (CL) based representation to the DEC-POMDP tuple (described in Section II-A). Firstly, we assume that the individual agent models, i.e., the model of an agent without other agents in the environment is available. This is not a strict assumption because understanding the coordination between agents in a domain requires understanding of individual agent models. Individual model for an agent i, $\mathcal{M}_i$, is represented using the tuple $\langle S_i, A_i, \Omega_i, \mathcal{P}_i, \mathcal{R}_i, \mathcal{O}_i, H \rangle$. Therefore, $\mathbb{S} = \times_i S_i, \mathbb{A} = \times_i A_i$ and $\Omega = \times_i \Omega_i$. In problems, where there are "global state" or "unaffected state" features, the state space of every agent will have those features. The dependencies between agents due to these features are captured using CLs (as described below).

Secondly, we assume all the interactions that could happen between agents are described in terms of coordination locales. A CL is defined as a tuple of states and actions for all the n agents involved in the interaction and is represented as $\langle e, (s_i)_1^n, (a_i)_1^n, \Gamma \rangle$. This set of CLs are obtained from the joint model and the individual models. A CL, $\langle e, (s_i)_1^n, (a_i)^n, \Gamma \rangle$ exists if either of the following conditions hold for any $e < H$:

$$\mathbb{P}^e((s_i)_1^n, (a_i)_1^n, (s_i')_1^n) \neq \times_{i \leq n} \mathcal{P}_i^e(s_i, a_i, s_i'), \Gamma = 0, \quad (1)$$

$$\mathbb{R}^e((s_i)_1^n, (a_i)_1^n, (s_i')_1^n)) \neq \oplus_{i \leq n} \mathcal{R}_i^e(s_i, a_i, s_i'), \Gamma = 0, \quad (2)$$

$$\mathbb{O}_k^e((s_i)_1^n, (a_i)_1^n, \omega_k) \neq \mathcal{O}_k^e(s_k, a_k, \omega_k), \forall k, \Gamma = 1 \quad (3)$$

CLs detected by using Equation 1, Equation 2 and Equation 3 are appropriately referred to as the Transition CLs, Reward CLs or Observation CLs. Γ is used to make the distinction between Transition/Reward CLs ($\Gamma = 0$)and Observation CLs ($\Gamma = 1$). We make the distinction between CLs, because states s_1, s_2 in Transition or Reward CLs refer to the source states, but in Observation CLs they refer to the destination states.

Since we assume a generic DEC-POMDP model where there are transition, observation and reward dependencies, the Future Time Coordination Locales (FTCLs) as introduced in Varakantham *et al.* [12] can be represented using STCLs. FTCLs arise because modification of global state by one agent can affect other agents. Because we consider a generic DEC-POMDP model, observation dependencies are present and hence FTCLs can be represented using STCLs. Since we only have STCLs, henceforth, we refer to them as CLs. Furthermore, we assume that n-ary interactions (CLs involving n-agents) can be modeled using multiple binary interactions (CLs with two agents). Thus, the shaping heuristics are provided with respect to two interacting agents.

IV. Generic TREMOR

Due to the generic nature of the problems being solved, unlike in TREMOR (or D-TREMOR), there is no role allocation at every iteration of the algorithm. Instead, the goal is to compute a joint policy that maximizes expected value. The core algorithm remains the same as the one described in Algorithm 1. The key differences are the enhancements that make GenTREMOR suitable for application to generic DEC-POMDPs are:

(i) Updated Model shaping heuristics to account for CL dependencies and observation dependence .

(ii) Computation of CL occurrence probability, $Pr_{i,cl}^0$.

A. Updated Model Shaping Heuristics

There are two key steps in which shaping of individual agent models occurs in TREMOR(and D-TREMOR) and consequently in GenTREMOR:

Step 1: Firstly, in updating the individual models to account for the effects of active CLs; and

Step 2: Secondly, depending on the utility of the CLs to the team as a whole, incentivizing (in terms of extra reward accrued by the team) or hindering (penalties incurred by the team) the occurrence of CLs in the agent models.

In evaluating a CL (line 7 of Algorithm 1), **Step 1** is employed. While on line 8, both steps are used in shaping of models. We provide new heuristics for shaping models in Step 1. This is an important modification which dictates the quality of the final joint policy. **Step 1** was performed in TREMOR using the following update expressions.

$Pr^0_{j,cl}$ represents the probability of occurrence of cl with respect to agent j, where $cl = \langle e, (s_i, s_j), (a_i, a_j), \Gamma\rangle$.

$$\mathbb{P}^e_{cl}(s_i, a_i, s'_i) \leftarrow \sum_{s' \in S: s'=(s'_i, s'_j)} \mathbb{P}((s_i, s_j), (a_i, a_j), (s'_i, s'_j))$$

$$\mathcal{P'}^e_i \leftarrow Pr^0_{j,cl} \cdot \mathbb{P}^e_{cl} + (1 - Pr^0_{j,cl}) \cdot \mathcal{P}^e_i \quad (4)$$

$$\mathbb{R}^e_{cl}(s_i, a_i) \leftarrow \mathbb{R}((s_i, s_j), (a_i, a_j))$$

$$\mathcal{R'}^e_i \leftarrow Pr^0_{j,cl} \cdot \mathbb{R}^e_{cl} + (1 - Pr^0_{j,cl}) \cdot \mathcal{R}^e_i \quad (5)$$

In the above expressions, the transition and reward functions were updated according to the probability of a CL being active. While there was no theoretical justification provided [12] on the suitability of the above update expressions, the above heuristics were shown to provide improved performance on loosely coupled disaster rescue problems.

However, in tightly coupled problems, the above update expressions can have undesirable effects. For instance, consider the scenario where there are two cls, cl1 and cl2 with the same e, s_i and a_i and different s_j and a_j. Now, if the model for agent i is updated corresponding to $cl1$ first and $cl2$ next, then the model update corresponding to $cl1$ could potentially be overwritten by the model update for $cl2$. To address such inconsistencies in model updates, we propose new model shaping heuristics. We use the set $CL^i_{s,a}$ to represent all CLs which have the same state s and action a corresponding to agent i. Instead of considering the occurrence and non occurrence of each CL separately, we aggregate corresponding to all CLs which have the same state and action pair for the specific agent. Considering this, the new shaping heuristics for shaping of transition and reward functions are as follows:

$$\tilde{\mathcal{P}}^e_i \leftarrow \sum_{cl \in CL^i_{s,a}} Pr^0_{j,cl} \cdot \mathbb{P}^e_{cl} + (1 - \sum_{cl \in CL^i_{s,a}} Pr^0_{j,cl}) \cdot \mathcal{P}^e_i \quad (6)$$

$$\tilde{\mathcal{R}}^e_i \leftarrow \sum_{cl \in CL^i_{s,a}} Pr^0_{j,cl} \cdot \frac{\mathbb{R}^e_{cl}}{2} + (1 - \sum_{cl \in CL^i_{s,a}} Pr^0_{j,cl}) \cdot \mathcal{R}^e_i \quad (7)$$

In these expressions, we compute new transition and reward values by accounting for affects of all the CLs (with same state, action pairs) at once and hence effects of a CL are not overwritten. Our model shaping also accounts for CLs related to observation dependencies and the shaping of observation probabilities is performed in a similar manner.

$$\mathbb{O}^e_{cl}(s_i, a_i, \omega_i)) \leftarrow \mathbb{O}_i((s_i, s_j), (a_i, a_j), \omega_i)$$

$$\tilde{\mathcal{O}}^e_i \leftarrow \sum_{cl \in CL^i_{s,a}} Pr^0_{j,cl} \cdot \mathbb{O}^e_{cl} + (1 - \sum_{cl \in CL^i_{s,a}} Pr^0_{j,cl}) \cdot \mathcal{O}^e_i \quad (8)$$

B. Computation of $Pr_{i,cl}$

In the computation of CL occurrence probability, $Pr_{i,cl}$, there are two key changes that we introduce in GenTREMOR. Firstly, we modify the procedure employed in TREMOR (and D-TREMOR) to compute $Pr_{i,cl}$, so that it is applicable to tightly coupled problems. The goal of model shaping is to enable the sum of expected reward computed using individual shaped models to be equal the joint value computation for the DEC-POMDP. When there are dependencies between CLs (a key characteristic of tightly coupled problems, as mentioned in the previous section), the computation of $Pr_{i,cl}$ as employed in TREMOR does not accomplish the model shaping goal. To address this, we introduce a modified procedure:

$$Pr^t_{i,cl}(b_i) = \begin{cases} b_i(s_i)\pi_i(b_i, a) & t = e\,, \\ \sum_{a \in A_i} \pi_i(b_i, a) \sum_{\omega \in \Omega_i} P^a_i(\omega|b_i) \cdot Pr^{(t+1)}_{i,cl}(G^{a,\omega,t}_i(b_i)) & t \leq e \end{cases},$$

$$G^{a,\omega,t}_i(b_i)(s') = \frac{\mathcal{O}_i(s', a, \omega) \sum_{s \in S} \mathcal{P}_i(s, a, s') b_i(s)}{P^a_i(\omega|b_i)}, \forall t > 0$$

$$G^{a,\omega,0}_i(b_i)(s') = \frac{\mathcal{O}_i(s', a, \omega)}{P^a_i(\omega|b_i)},$$

$$P^a_i(\omega|b_i) = \sum_{s' \in S} [\mathcal{O}_i(s', a, \omega) \sum_{s \in S} \mathcal{P}_i(s, a, s') b_i(s)]$$

The key change from the computation employed in TREMOR (described in Section II) is the introduction of a separate expression for $G^{a,\omega}(b^0)(s')$ to account for the dependencies between CLs at different decision epochs which lead to b^0 getting multiplied many times. By introducing this factor, we are discounting such computations. As we will demonstrate in Section V, this leads to equivalence of individual expected rewards and joint expected reward.

Secondly, the computation of occurrence probability for an Observation CL is different from that of a Transition or Reward CL. As mentioned earlier, this is because in an Observation CL, $\langle e, (s_i, s_j), (a_i, a_j), 1\rangle$, s_i and s_j refer to destination states from taking the action a_i and a_j respectively. However for Transition and Reward CLs, they refer to source states from which the actions are being taken. To that end, we update the procedure as follows:

$$Pr^t_{i,cl}(b_i) = \begin{cases} b_i(s_i)\pi_i(b_i, a) & t = e, \Gamma = 0, \\ \pi_i(b_i, a) * \sum_{\tilde{s}_i} b_i(\tilde{s}_i)\mathcal{P}(\tilde{s}_i, a, s_i) & t = e, \Gamma = 1, \\ \\ \sum_{a \in A_i} \pi_i(b_i, a) \sum_{\omega \in \Omega_i} P^a_i(\omega|b_i) \cdot Pr^{(t+1)}_{i,cl}(G^{a,\omega,t}_i(b_i)) & t \leq e \end{cases}$$

V. Theoretical Results

In this section, we provide theoretical justification on why the model shaping performed in **Step 1** using the new heuristics of Equations 6 and 7 is accurate. We prove the accuracy of **Step 1** when only transition and reward dependencies are present. While, this does not prove the general case, it is a first step towards understanding shaping heuristics and for developing formal procedures to construct new heuristics.

Henceforth, we will use the following notation corresponding to $cl = \langle \tau - 1, (s_1, s_2), (a_1, a_2), 0\rangle$:

$$I^{\tau-1}_{cl} \leftarrow I_{\langle \tau-1,(s_1,s_2),(a_1,a_2),0\rangle \in CLs}$$
$$I^{\tau-1}_{\neg cl} \leftarrow I_{\langle \tau-1,(s_1,s_2),(a_1,a_2),0\rangle \notin CLs}$$

where I_{cond} is 1 if $cond$ is true and 0 otherwise.

$$\begin{aligned} Pr^0_{i,cl} &\leftarrow Pr^{\tau-1}_{\pi_i}(s^{\tau-1}_i, a^{\tau-1}_i) \\ \mathbb{T}^{\tau-1}_{cl} &\leftarrow \mathbb{T}^{\tau-1}((s^{\tau-1}_1, s^{\tau-1}_2), (a_1, a_2), (s^\tau_1, s^\tau_2)) \\ \mathcal{T}^{\tau-1}_i &\leftarrow \mathcal{T}_i(s^{\tau-1}_i, a^{\tau-1}_i, s^\tau_i) \\ \mathbb{O}^{\tau-1}_{cl} &\leftarrow \mathbb{O}^{\tau-1}((s^\tau_1, s^\tau_2), (a^{\tau-1}_1, a^{\tau-1}_2), (\omega^\tau_1, \omega^\tau_2)) \\ \mathbb{O}^{\tau-1}_{i,cl} &\leftarrow \mathbb{O}^{\tau-1}_i((s^\tau_i, s^\tau_j), (a^{\tau-1}_i, a^{\tau-1}_j), \omega^\tau_i) \\ \mathcal{O}^{\tau-1}_i &\leftarrow \mathcal{O}^{\tau-1}_i(s^\tau_i, a^{\tau-1}_i, \omega^\tau_i) \\ \mathbb{R}^{\tau-1}_{cl} &\leftarrow \mathbb{R}^{\tau-1}((s^{\tau-1}_1, s^{\tau-1}_2), (a^{\tau-1}_1, a^{\tau-1}_2), (s^\tau_1, s^\tau_2)) \\ \mathcal{R}^{\tau-1}_i &\leftarrow \mathcal{R}^{\tau-1}_i(s^{\tau-1}_i, a^{\tau-1}_i) \end{aligned}$$

For the two agent case, the value for a joint policy, π is computed as follows:

$$\begin{aligned} V_{(\pi_1,\pi_2)}((b^{\tau-1}_1, b^{\tau-1}_2)) = ER((b^{\tau-1}_1, b^{\tau-1}_2), (\pi_1.a, \pi_2.a)) + \\ \sum_{s^\tau_2, \omega^\tau_1, \omega^\tau_2}^{s^{\tau-1}_1, s^{\tau-1}_2, s^\tau_1} \mathbb{T}^{\tau-1} \cdot \mathbb{O}^{\tau-1} \cdot b_1(s^{\tau-1}_1) \\ \cdot b_2(s^{\tau-1}_2) \cdot V_{(\pi_1(\omega^\tau_1), \pi_2(\omega^\tau_2))}((s^\tau_1, s^\tau_2)) \end{aligned} \tag{9}$$

For each of the agents ($i \in \{1, 2\}$), the value of a policy (without considering the other agent) is computed as follows:

$$V_{\pi_i}(b^{\tau-1}_i) = ER(b^{\tau-1}_i, \pi_i.a) + \sum_{s^\tau_i, s^{\tau-1}_i}^{\omega^\tau_i} \mathcal{T}^{\tau-1}_i \cdot \mathcal{O}^{\tau-1}_i \cdot b^{\tau-1}_i \cdot V_{\pi_i(\omega^\tau_i)}(s^\tau_i)$$

The heuristics for computing new transition ($\hat{\mathcal{T}}_i$) and reward ($\hat{\mathcal{R}}_i$) and observation($\hat{\mathcal{O}}_i$) values are as follows:

$$\begin{aligned} \hat{\mathcal{T}}^{\tau-1}_1 &= \sum_{cl} Pr^0_{2,cl} \mathbb{T}^{\tau-1}_{cl} + (1 - \sum_{cl} Pr^0_{2,cl}) \mathcal{T}^{\tau-1}_1 \\ &= \sum_{s^{\tau-1}_2, s^\tau_2}^{a^{\tau-1}_2} Pr^{\tau-1}_{\pi_2}(s^{\tau-1}_2, a^{\tau-1}_2)[I^{\tau-1}_{cl} \cdot \mathbb{T}^{\tau-1}_{cl} + I^{\tau-1}_{\neg cl} \cdot \mathcal{T}^{\tau-1}_1] \end{aligned} \tag{10}$$

$$\begin{aligned} \hat{\mathcal{R}}^{\tau-1}_1 &= \sum_{cl} Pr^0_{2,cl} \frac{\mathbb{R}^{\tau-1}_{cl}}{2} + (1 - \sum_{cl} Pr^0_{2,cl}) \mathcal{R}^{\tau-1}_1 \\ &= \sum_{s^{\tau-1}_2}^{a^{\tau-1}_2} Pr_{\pi_2}(s^{\tau-1}_2, a^{\tau-1}_2)[I^{\tau-1}_{cl} \cdot \frac{\mathbb{R}^{\tau-1}_{cl}}{2} + I_{\neg cl} \cdot \mathcal{R}^{\tau-1}_1] \end{aligned} \tag{11}$$

For ease of explanation, we provide the theoretical justification for two agents, however, it naturally extends to multiple agents.

Proposition 1: Given policies π_1 and π_2 of two agents, Equations 10 and 11 provide for accurate compution of joint value, i.e.,

$$V_{\pi_1,\pi_2}(b^{\tau-1}_1, b^{\tau-1}_2) = \hat{V}_{\pi_1}(b^{\tau-1}_1) + \hat{V}_{\pi_1}(b^{\tau-1}_1) \tag{12}$$

where $i \in \{1, 2\}$ and $\hat{V}_{\pi_i}(b^{\tau-1}_i)$ represents the value function corresponding to agent i's shaped model.

Proof. We employ mathematical induction over time horizon to prove this proposition.
Base Case: Time Horizon, $H = 1$
Since time horizon is 1, we show that the joint policy immediate reward is equal to the sum of individual policy immediate rewards with model shaping. The expression for computing joint policy value is as follows:

$$\begin{aligned} V_\pi(b^0) &= ER(b^0, (\pi_1.a, \pi_2.a)) \\ &= \sum_{(s^0_1, s^0_2)} b^0_1(s^0_1) b^0_2(s^0_2)\{I^0_{cl} \cdot \mathbb{R}^0_{cl} + I_{\neg cl} \cdot [\mathcal{R}^0_1 + \mathcal{R}^0_2]\} \end{aligned} \tag{13}$$

The value for agent 1 due to shaping of reward function is:

$$\hat{V}_{\pi_1}(b^0_1) = \sum_{s^0_1} b^0_1(s^0_1)\hat{\mathcal{R}}^0_1$$

Substituting expression for $\hat{\mathcal{R}}^0_1$ from Equation 11

$$= \sum_{s^0_1} b^0_1(s^0_1) \sum_{s^0_2} \{Pr^0(s^0_2, \pi_2.a)\{I^0_{cl} \cdot \frac{\mathbb{R}^0_{cl}}{2} + I^0_{\neg cl}\mathcal{R}^0_1\}$$

Since $Pr^0(s_2^0, \pi_2.a) = b_2^0(s_2^0)$ for T = 0

$$= \sum_{(s_1^0, s_2^0)} b_1^0(s_1^0) b_2^0(s_2^0)\{I_{cl}^0 \cdot \frac{\mathbb{R}_{cl}^0}{2} + I_{\neg cl}\mathcal{R}_1^0\} \tag{14}$$

By summing Eqn 14 for both agents, we obtain the expression in Equation 13. Therefore, it is proved for problems with horizon = 1.

Inductive Step: Now, we assume the proposition holds for problems with horizon = τ, and prove it for horizon = $\tau+1$. The joint value function for horizon = $\tau+1$ (extending from Equation 9) is:

$$\begin{aligned} V_\pi(b^0) = ER(b^0, (\pi_1.a, \pi_2.a)) + \\ \sum_{s_1^1, s_2^1, s_1^0, s_2^0, \omega_1^1, \omega_2^1} \{I_{cl}^0 \cdot \mathbb{T}_{cl}^0 + I_{\neg cl}^0 \cdot \mathcal{T}_1^0 \cdot \mathcal{T}_2^0\} \cdot \mathbb{O}^0 \\ \cdot b_1^0(s_1^0) \cdot b_2^0(s_2^0) \cdot V_{\pi_1(\omega_1^1), \pi_2(\omega_2^1)}(s_1^1, s_2^1) \end{aligned} \tag{15}$$

The first part of RHS in Equation 15 can be divided into two value components one for each agent as proved above for the case when horizon is 1. Therefore, we will only focus on showing that the second part of RHS in Equation 15 can be expressed as sum of value components one for each agent. We begin from the value function of agent 1:

$$\hat{V}_{\pi_1}(b_1^0) = ER(b_1^0, \pi_1.a) + \sum_{s_1^1, s_1^0, \omega_1^1} \hat{\mathcal{T}}_1^0 \cdot \hat{\mathcal{O}}_1^0 \cdot b_1^0(s_1^0) \cdot V_{\pi_1(\omega_1^1)}(s_1^1)$$

Substituting Equation 10,

$$\begin{aligned} = ER(b_1^0, \pi_1.a) + \sum_{s_1^1, s_1^0, \omega_1^1} \sum_{s_2^0} Pr(s_2^0, \pi_2.a) \\ \cdot \{I_{cl}^0 \cdot \sum_{s_2^1} \mathbb{T}_{cl}^0 + I_{\neg cl}^0 \cdot \mathcal{T}_1^0\} \cdot \mathcal{O}_1^0 \cdot b_1^0(s_1^0) \cdot V_{\pi_1(\omega_1^1)}(s_1^1) \end{aligned}$$

Since $\sum_{s_2^1} \mathcal{T}_2^0 = 1$,

$$\begin{aligned} &= ER(b_1^0, \pi_1.a) + \sum_{s_1^1, s_1^0, \omega_1^1} \sum_{s_2^0} b_2^0(s_2^0) \cdot b_1^0(s_1^0) \\ &\{I_{cl}^0 \cdot \sum_{s_2^1} \mathbb{T}_{cl}^0 + I_{\neg cl}^0 \cdot \mathcal{T}_1^0 \cdot \sum_{s_2^1} \mathcal{T}_2^0\} \cdot \mathcal{O}_1^0 \cdot V_{\pi_1(\omega_1^1)}(s_1^1) \\ &= ER(b_1^0, \pi_1.a) + \sum_{s_1^1, s_1^0, \omega_1^1, s_2^0, s_2^1} b_2^0(s_2^0) \cdot b_1^0(s_1^0) \\ &\{I_{cl}^0 \cdot \mathbb{T}_{cl}^0 + I_{\neg cl}^0 \cdot \mathcal{T}_1^0 \cdot \mathcal{T}_2^0\} \cdot \mathcal{O}_1^0 \cdot V_{\pi_1(\omega_1^1)}(s_1^1) \end{aligned}$$

Substituting expression for (i) probability of occurrence of all state, action pairs ($\sum_{\omega_2^1} O_2^0$) at second decision epoch given the policy and initial belief state in (ii) providing expression for future value, $V_{\pi_1(\omega_1^1)}(s_1^1)$ to account for CLs at second decision epoch,

$$\begin{aligned} = ER(b_1^0, \pi_1.a) + \sum_{s_1^1, s_1^0, \omega_1^1, s_2^0, s_2^1} b_2^0(s_2^0) \cdot b_1^0(s_1^0) \\ \{I_{cl}^0 \cdot \mathbb{T}_{cl}^0 + I_{\neg cl}^0 \cdot \mathcal{T}_1^0 \cdot \mathcal{T}_2^0\} \cdot \mathcal{O}_1^0 \cdot \\ \sum_{\omega_2^1} O_2^0 \cdot V_{\pi_1(\omega_1^1), \pi_2(\omega_2^1)}^1(s1, s2) \end{aligned} \tag{16}$$

The new future value function, $V_{\pi_1(\omega_1^1), \pi_2(\omega_2^1)}^1$ represents value of agent 1 given policy trees for horizon τ given the observations ω_1^1 and ω_2^1 at the first decision epoch. We first substitute the assumption due to mathematical induction in Equation 15 for the future value term for a state. Then, by summing Equation 16 for both agents, we obtain the updated Equation 15. Hence proved. ■

VI. Experimental Results

D-TREMOR scales with respect to number of agents (up to 100 agents), time horizon and state space in domains with sparse interactions between agents. Since the core algorithm and the approximations are the same as in D-TREMOR, GenTREMOR exhibits similar scale up and behavior. In fact, on rescue problems introduced in Prasanna *et al* [13], we are able to reproduce similar performance as D-TREMOR (as illustrated in Figure II-B). Our focus in this section is to evaluate GenTREMOR's performance(with respect to runtime and solution quality) on standard DEC-POMDP benchmark problems, where agents are tightly coupled.

We consider four of the standard DEC-POMDP benchmark problems, namely the tiger [9], multi-agent channel broadcast [11], box pushing [11] and finally the meeting of agents on a three by three grid [1]. We mention the number of joint states ($|S|$), individual agent actions ($|A_i|$)and individual agent observations ($|\Omega_i|$) below the name of the problem in Tables I and VI. Furthermore, in all the four problems, agents are tightly coupled due to transition, observation and reward dependencies.

Meeting on a grid $\langle 81, 4, 9\rangle$		
H	GenTREMOR	Existing(#)
3	0.131	0.133 (*)
4	0.429	0.433 (*)
5	0.893	0.896 (*)
10	4.647	3.85
100	93.77	92.12

Table II
Comparison on the Meeting in a grid (3x3) problem.

While, there have been many algorithms that have been developed to solve DEC-POMDPs, we focus on some of the

	Horizon	3	4	5	6	7	8	9	10	20	50	100
MABC	GenTREMOR	2.91	3.81	4.71	5.61	6.51	7.41	8.31	9.21	18.21	45.21	90.21
$\langle 2, 3, 2\rangle$	Existing Best(#)	2.99	3.89	4.79	5.69	6.59	7.49	8.39	9.29	18.29	45.29	90.29
Tiger	GenTREMOR	5.18	4.8	7.4	11.07	10.82	12.77	16.26	16.57	27.74	55.56	100.25
$\langle 2, 3, 2\rangle$	Existing Best(#)	5.19	4.8	5.38	9.91	9.67	9.42	12.57	13.49	-	-	93.24
Box Pushing	GenTREMOR	66.1	93.2	102.2	112.8	123.2	149.2	159.6	179.4	366.3	907.8	1810.9
$\langle 100, 4, 5\rangle$	Existing Best(#)	66.1	98.6	100.6	117.6	133.1	182.9	187.0	189.3	415.2	1051.8	2112.1

Table I
COMPARISON OF GENTREMOR AGAINST EXISTING APPROACHES ON THE MABC, TIGER AND BOX PUSHING PROBLEMS. (#) BEST KNOWN RESULT OF APPLYING EITHER OF IMBDP, PBIP-IPG, MBDP-OC.

leading algorithms: IMBDP [11], PBIP-IPG [1] and MBDP-OC [4]. We primarily compare solution quality obtained by GenTREMOR and these competing algorithms. We do not have access to the algorithm implementations for the above mentioned algorithms, hence we compare against quality values reported in the publications. Similar to PBIP-IPG [1], we also employ a runtime cut-off of 10 CPU hours[1].

The maximum time horizon considered was 100. There is stochasticity involved in computation of a GenTREMOR joint policy, hence we further average over 50 initial seeds. In addition, the expected solution quality corresponding to a GenTREMOR joint policy is computed by simulating on the underlying DEC-POMDP model. For each of the problems, we averaged over 10000 simulations. The $MAX\_ITERATIONS$ in Algorithm 1 was set to 10. Table I and Table VI provide the comparison of solution quality. Since, not all approaches have reported on all the four benchmark problems, we compare against the best reported value amongst all approaches[2]. Here are the main results:

- GenTREMOR took less than two minutes on all problems when the time horizon was less than or equal to 10. For larger horizons, GenTREMOR finished much within the pre-specified cutoff time of 10 hours for all the problems.
- **MABC** : while IMBDP provides the best quality values (Table I), GenTREMOR provides quality values which are very close to it even for a horizon of 100.
- **Tiger**: GenTREMOR obtained optimal policies until horizon 4 and consistently outperformed existing approaches irrespective of the time horizon employed.
- **Box Pushing**: We obtained optimal solution quality for horizon 3 and performed better than existing approaches at horizon 5. However, for all other horizons, IMBDP out performed GenTREMOR.
- **Meeting on a 3x3 grid**: We obtained optimal solution quality for horizon less than or equal to five. Not only that, GenTREMOR outperformed existing approaches for other horizon values[3].

These results emphasize our claim that GenTREMOR not only exploits structure in CLs, but also is suited for computing joint policies in tightly coupled problems.

VII. RELATED WORK

We will now discuss relevance of approaches – other than social model shaping – employed for solving DEC-POMDPs. Becker et al (2004) provided approaches for solving transition independent DEC-MDPs. ND-POMDPs [10], [8], [7] extend transition-independence with network structure interactions. TD-POMDPs [14] exploit structure in transition function. While these approaches have improved the scalability considerably, they only solve a sub-class of DEC-POMDPs.

Guestrin *et al.* [6] have provided approaches for exploiting weak dependencies between agents for solving multiple MDP systems. The key difference from our work is their assumption of full communication between agents.

Seuken et al. [11] provide a memory bounded dynamic programming (MBDP) technique with linear space complexity for solving general DEC-POMDPs. Point Based Incremental Pruning (PBIP) by Dibangoye *et al.* [5] extends MBDP by replacing exhaustive backup with a branch and bound search in the space of joint policy trees. Amato *et al.* [1] extend PBIP to efficiently perform dynamic programming backups. While all the above mentioned leading approaches for solving DEC-POMDPs are able to solve problems with long horizons, they have been primarily limited to two agent problems even when there is sparse coordination. On the other hand, GenTREMOR is able to solve problems with reasonably long horizons (upto 100) as well as exploit sparse coordination to scale to 100 agent problems.

REFERENCES

[1] C. Amato, J. S. Dibangoye, and S. Zilberstein. Incremental policy generation for finite-horizon dec-pomdps. In *International Conference on Automated Planning and Scheduling*, 2009.

[1]All our experiments were run on a Intel Core 2 Duo 2.40 GHz CPU with 3 GB RAM

[2]Existing approaches provided optimal solutions for $H <= 4$

[3]We provided results for this example in a separate table, because we do not have results for benchmark algorithms on all the time horizons.

[2] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving Transition Independent Decentralized Markov Decision Processes. *JAIR*, 22:423–455, December 2004.

[3] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002.

[4] A. Carlin and S. Zilberstein. Value-based observation compression for dec-pomdps. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2008.

[5] J. S. Dibangoye, A. Mouaddib, and B. Chaib-draa. Pointbased incremental pruning heuristic for solving finite-horizon dec-pomdps. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2009.

[6] C. Guestrin and G. Gordon. Distributed planning in hierarchical factored mdps. In *Uncertainty in Artificial Intelligence*, 2002.

[7] A. Kumar and S. Zilberstein. Event-detecting multi-agent mdps: Complexity and constant-factor approximation. In *International Joint Conference on Artificial Intelligence*, 2009.

[8] J. Marecki, T. Gupta, P. Varakantham, M. Yokoo, and M. Tambe. Not all agents are equal: Scaling up distributed pomdps for agent networks. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2008.

[9] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *International Joint Conference on Artificial Intelligence*, 2003.

[10] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *American Association of Artificial Intelligence*, 2005.

[11] S. Seuken and S. Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Uncertainty in Artificial Intelligence*, 2007.

[12] P. Varakantham, J. Y. Kwak, M. Taylor, J. Marecki, P. Scerri, and M. Tambe. Exploiting coordination locales in distributed pomdps via social model shaping. In *International Conference on Automated Planning and Scheduling*, 2009.

[13] P. Velagapudi, P. Varakantham, P. Scerri, and K. Sycara. Distributed model shaping for scaling to decentralized pomdps with hundreds of agents. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2011.

[14] S. J. Witwicki and E. H. Durfee. Influence-based policy abstraction for weakly-coupled dec-pomdps. In *International Conference on Automated Planning and Scheduling*, 2010.